Session Code: TLS311

tools & languages

Visual C++ Whidbey: Leveraging Native Code with .NET

Scott Currie
Program Manager
Microsoft Visual C++
srcurrie@microsoft.com

PDC03

Make the connection

Agenda

- Native and .NET Interop
- Compilation Modes
- Interop Details
- __clrcall Calling Convention
- Mixed DLL Loading
- Delayed CLR Loading
- Application Domains

COM Interop

- .NET code calls to COM
  - .NET wrapper for COM code (tlbimp)
- COM code calls to .NET
  - COM wrapper for .NET code (tlbexp)
- Visual Studio provides interface
- CLR handles necessary type marshalling
  - This can be expensive
- Interop classes are produced
- Very useful if you have existing COM interfaces

COM Interop

demo

PDC03

Make the connection

DLL Import

- .NET calls to machine code only
- Use attribute in source code to specify
- Imports specified native DLL export
- Marshalling behavior can be explicitly defined
- Very useful for existing DLLs using exports

DLL Import

demo

PDC03

Make the connection

CLR Hosting

- Native calls to .NET code only
- Native code uses CLR Hosting API
  - ICLRRuntimeHost
- Native app must drive .NET code execution
- Not recommended for most users

CLR Hosting

demo

PDC03

Make the connection

C++ CLR / Interop Features

- Extremely Flexible
  - Add .NET code calls directly to existing C++ code
  - Call existing C++ code directly from .NET code
  - Call existing C++ code through .NET wrapper class
- Compile appropriate source with /clr
- Continue to call C++ code from native
  - No extra binaries
- Useful for almost all user scenarios

C++ CLR / Interop Features

demo

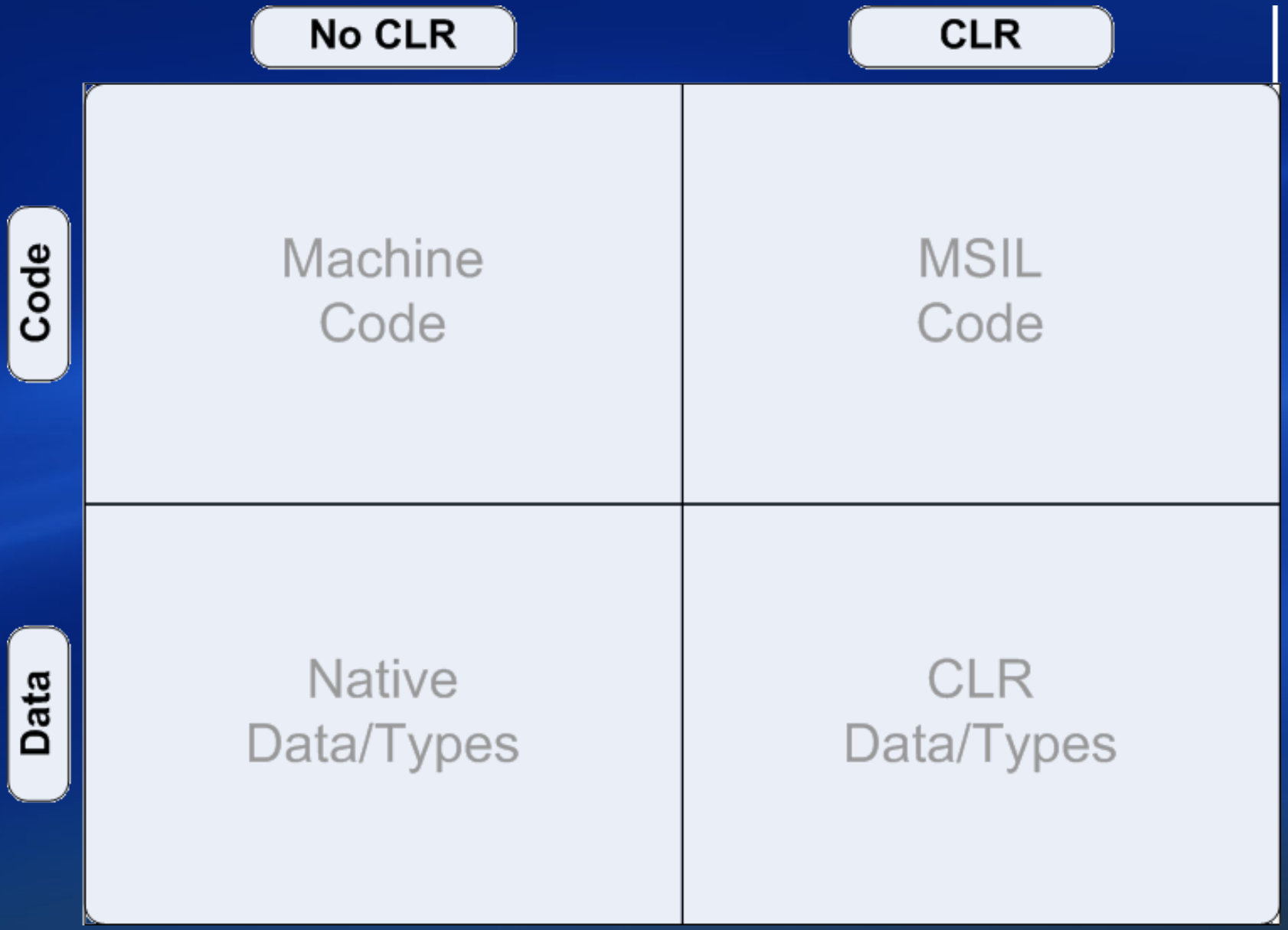
PDC03

Make the connection

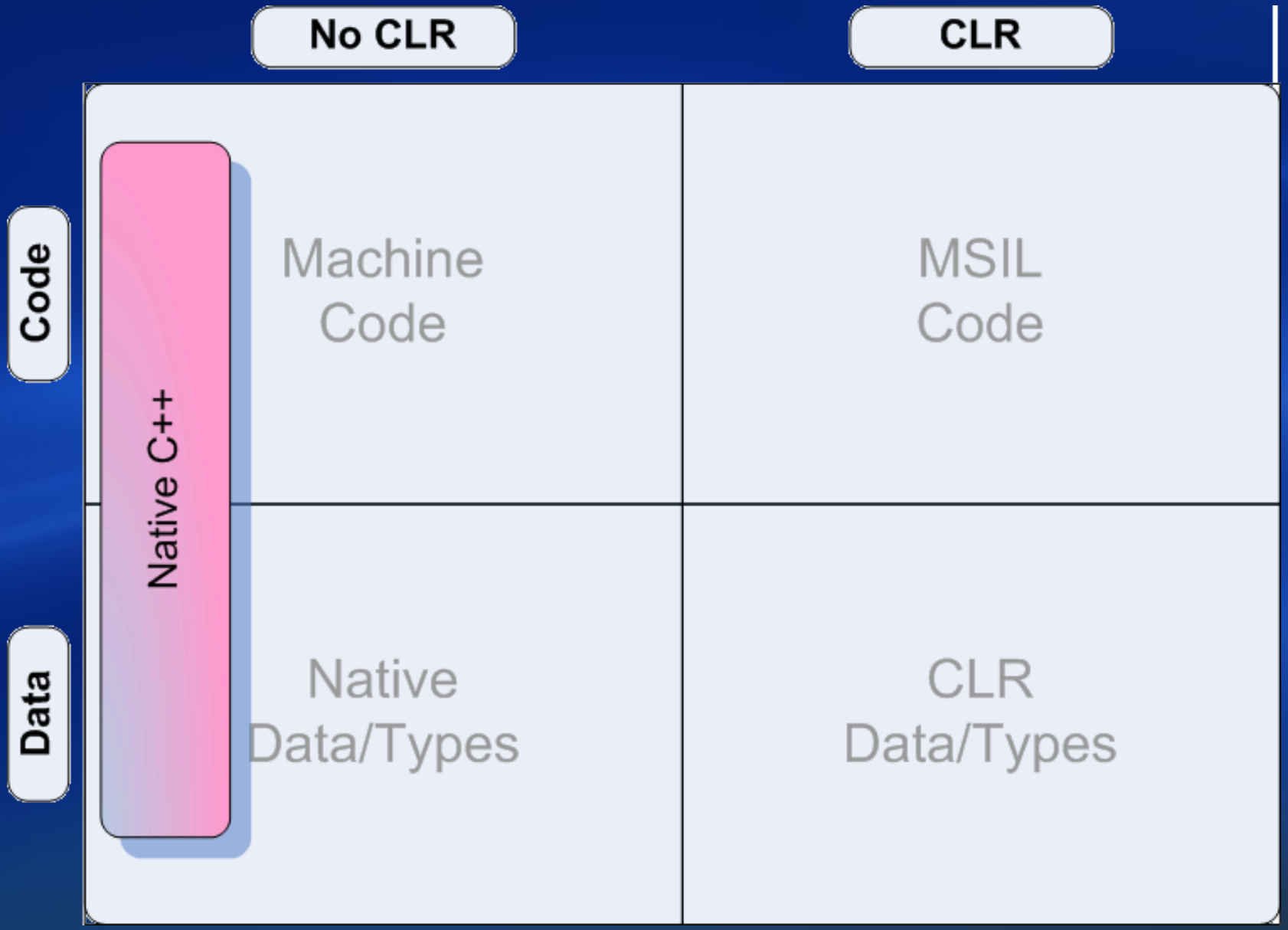
Agenda

- Using Native Code from .NET
- Compilation Modes
- Magic of IJW
- Managed Calling Convention
- Application Domains

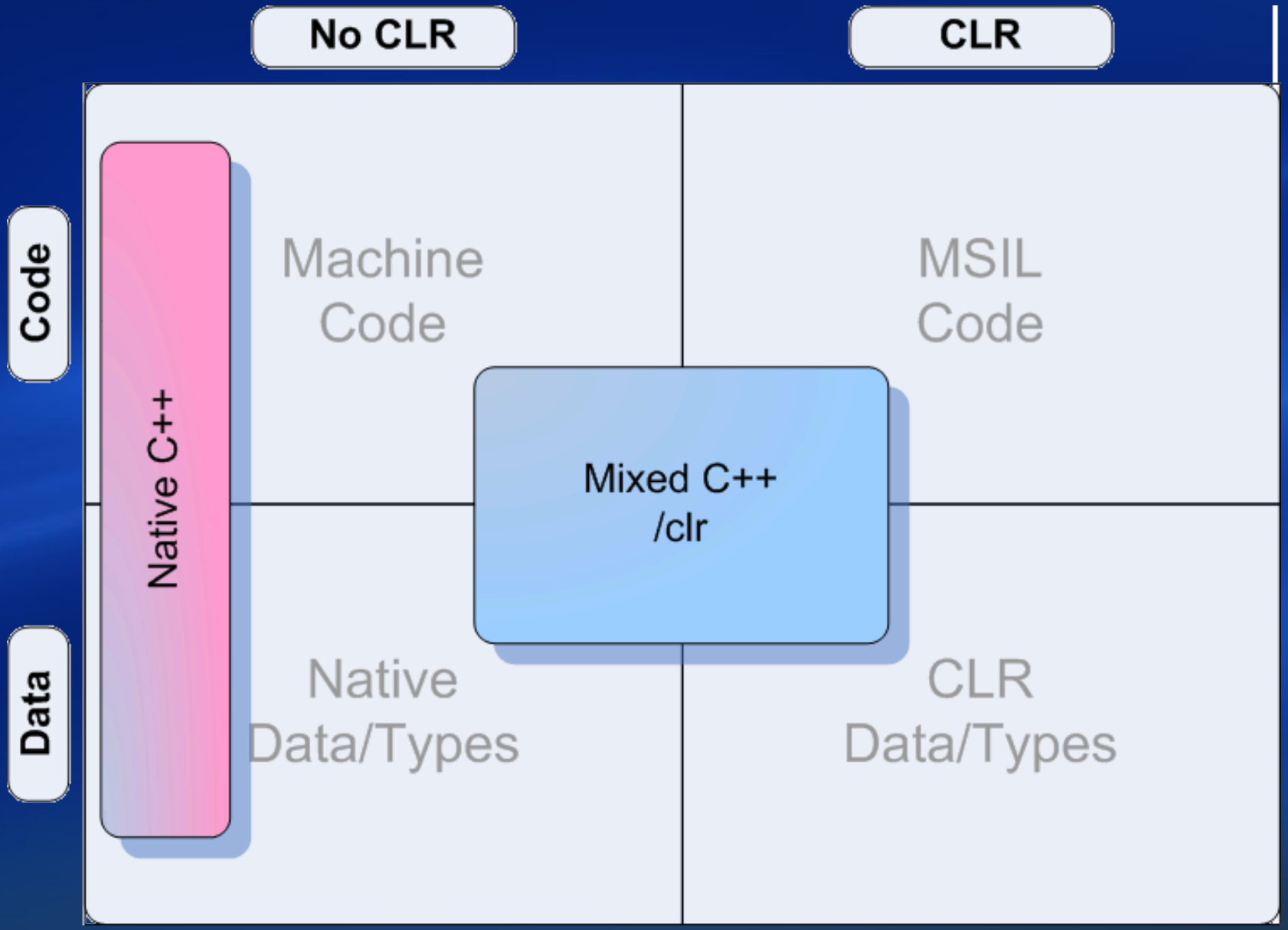
Compilation Modes



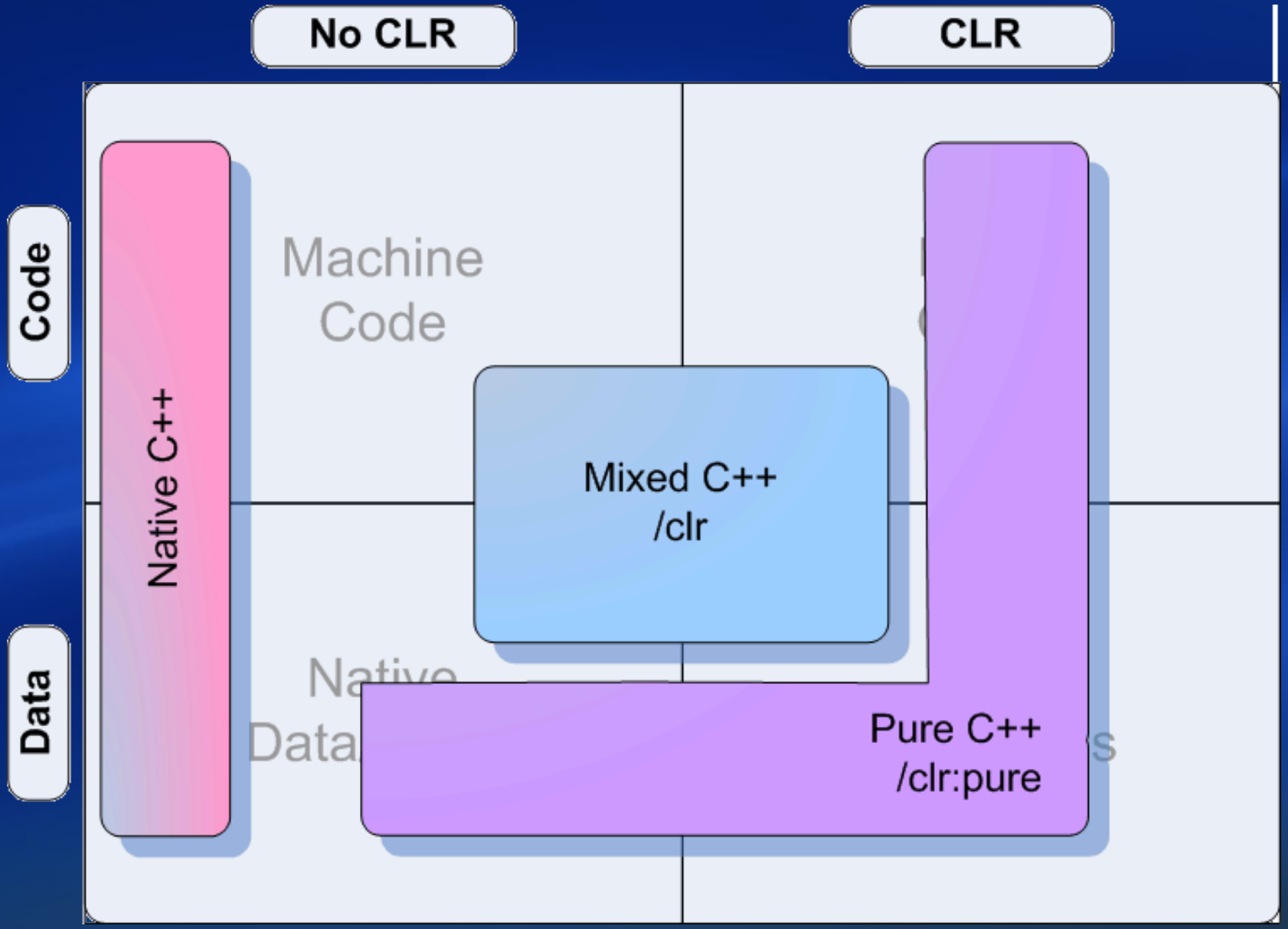
Compilation Modes



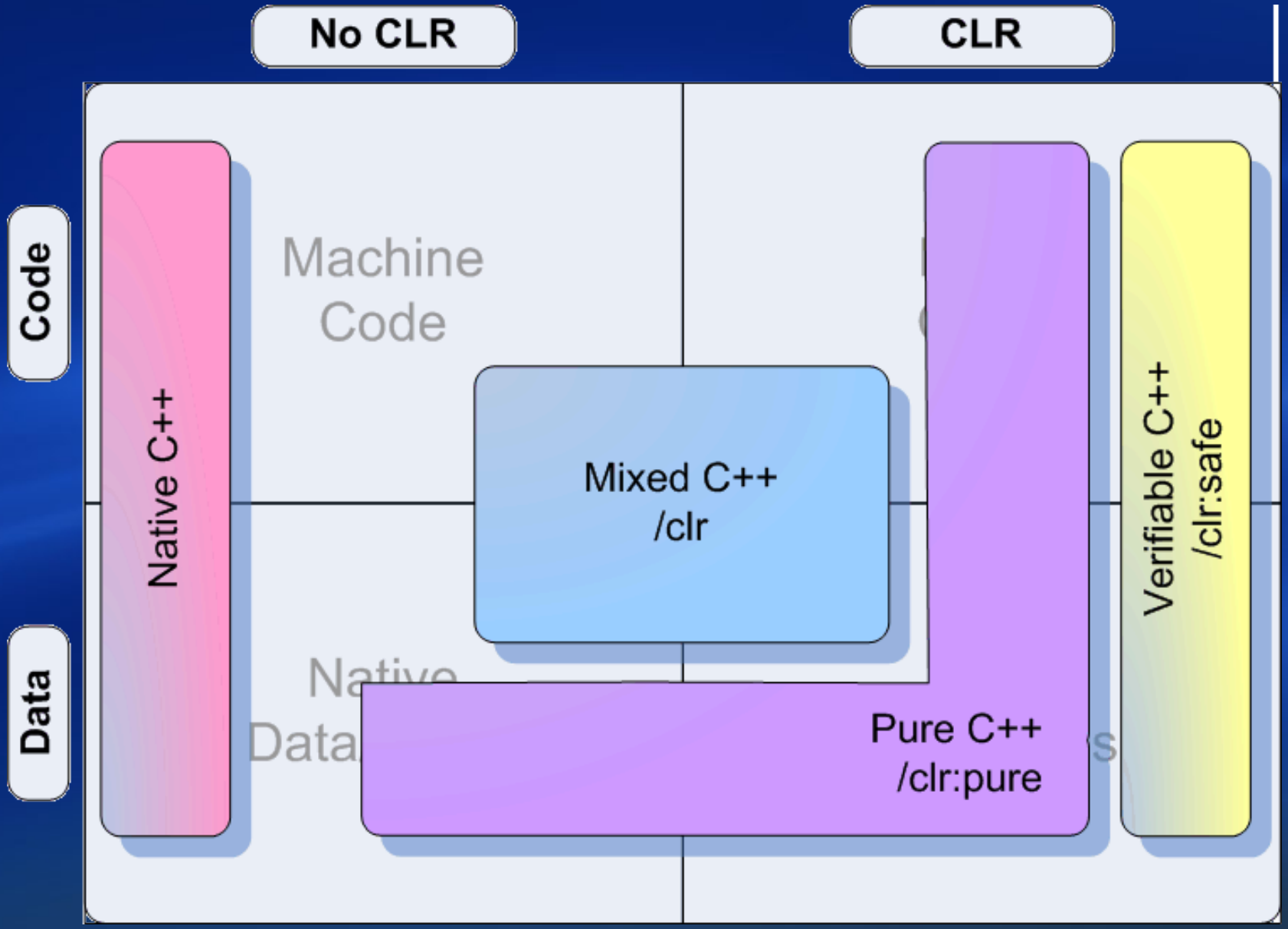
Compilation Modes



Compilation Modes



Compilation Modes



Agenda

- Native and .NET Interop
- Compilation Modes
- Interop Details
- __clrcall Calling Convention
- Mixed DLL Loading
- Delayed CLR Loading
- Application Domains

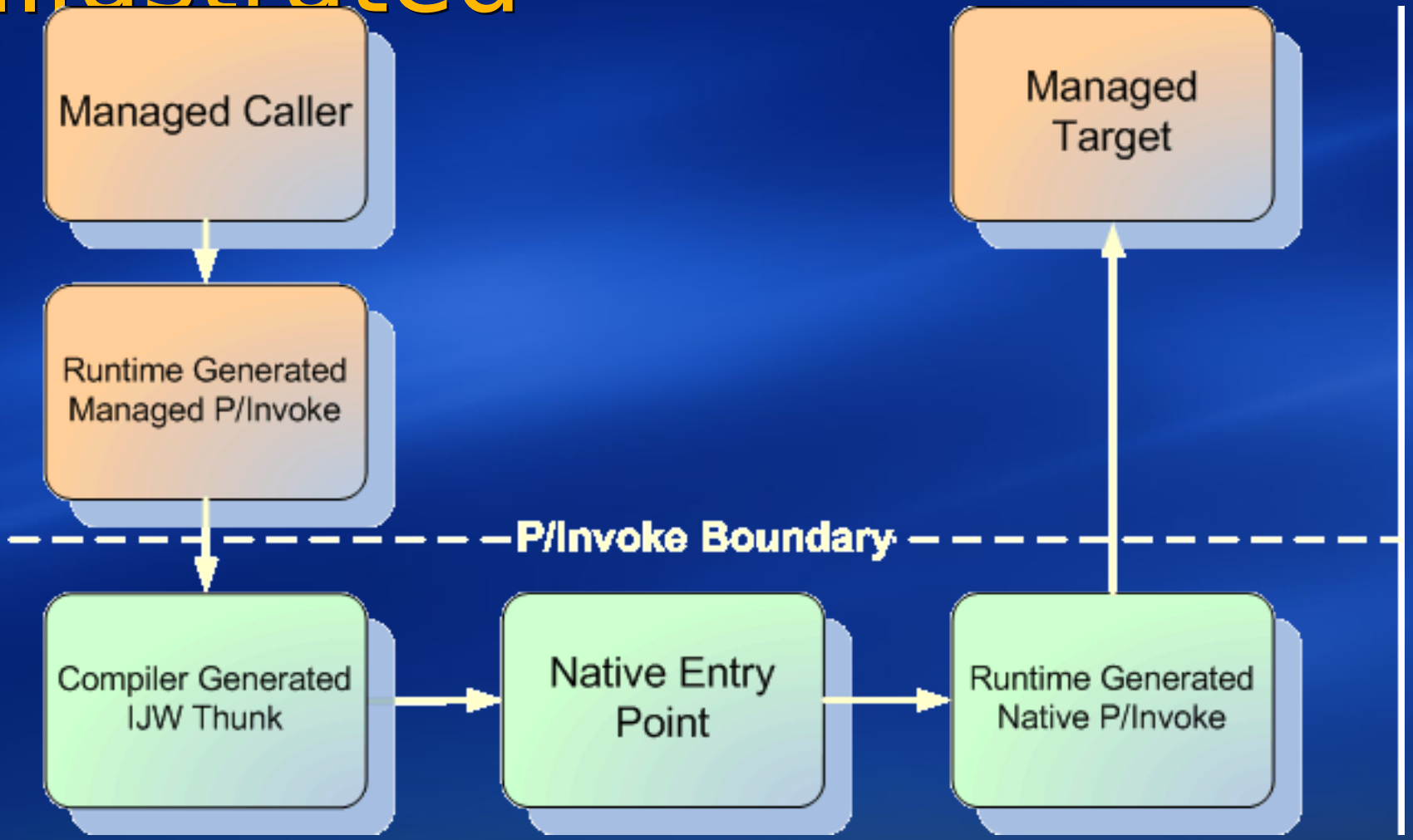
Interop Details

- All global/native member methods have both unmanaged and managed entry points
  - Actual method implementation
  - Forwarding thunk
- Method Implementation is MSIL unless:
  - Compiler can’t generate MSIL
    - Inline ASM
    - CPU specific intrinsics
  - User specifies machine code

Consequences

- Two entry points for each function
- One of the two must be statically chosen
- What if we lack information?
  - Function pointers (indirect calls)
  - VTable
- For compatibility, always choose unmanaged entry point
- Double-P/Invoke Problem

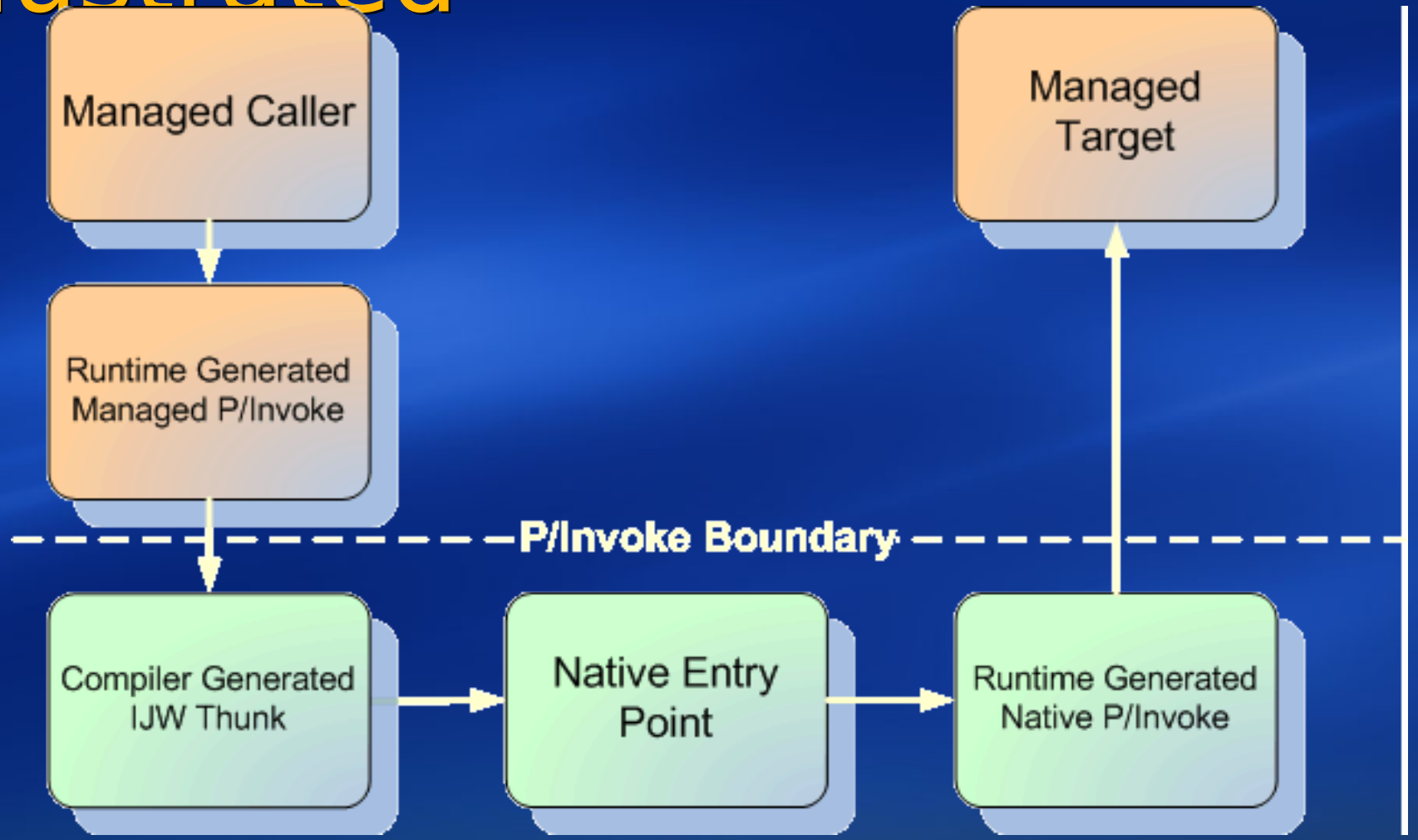
Double-P/Invoke Illustrated



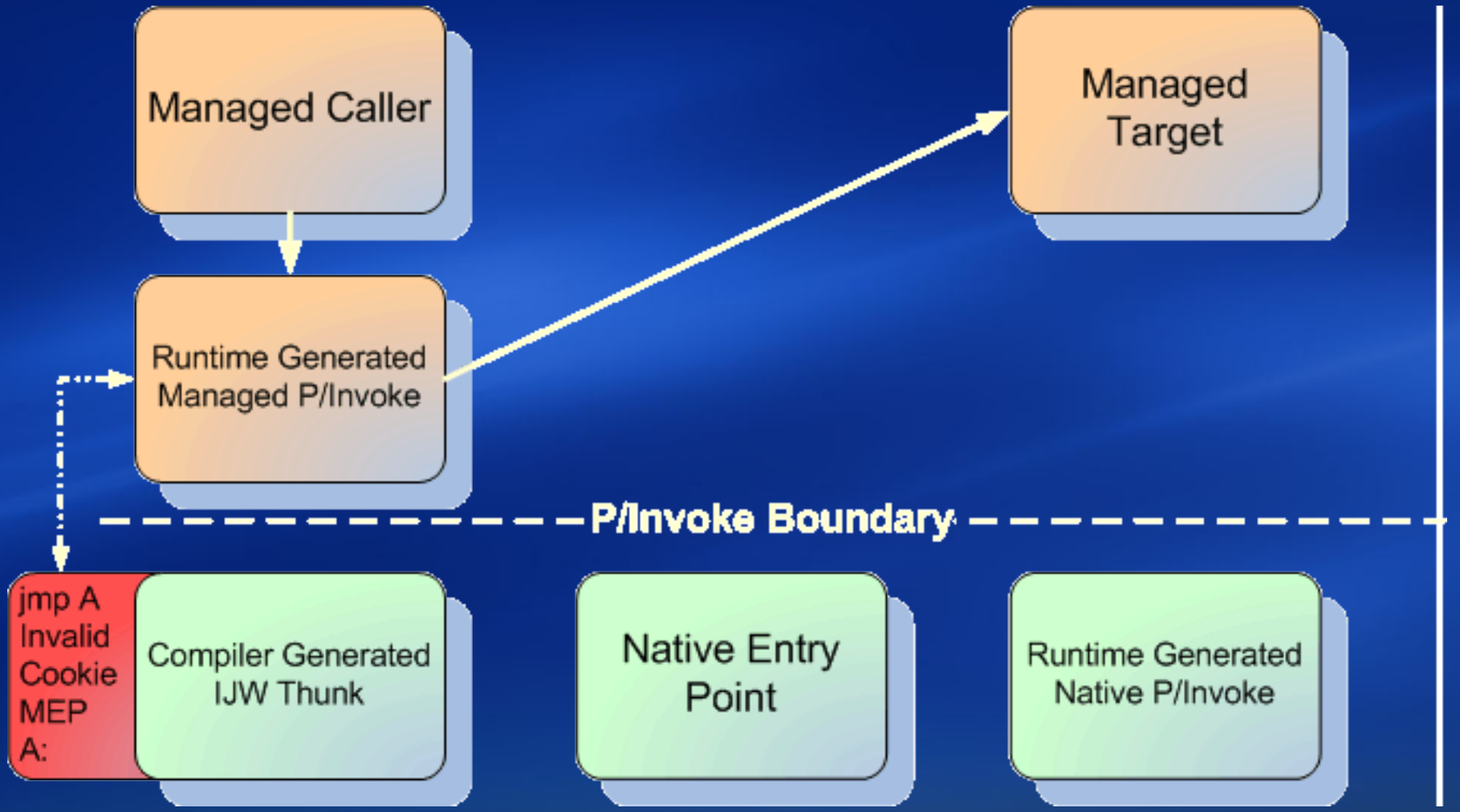
Double-P/Invoke Solutions

- Use /clr:pure
- Use __clrcall on target
- Do nothing → Auto-DP Elimination
  - Compiler emits thunks with cookies in them
    - 1st instruction jumps to thunk start for compatibility
    - 2nd instruction is invalid opcode
    - 3rd instruction contains magic cookie
    - 4th instruction contains address of Managed EP
  - Old runtimes unaffected
  - New runtimes
    - Check for cookie
    - If exists, jump directly to Managed EP without P/Invoke

Double-P/Invoke Illustrated



Auto-DP Elimination Illustrated



Moral Of The Story

- P/Invokes are expensive
- You have to think about the cost of transition thunks
- Design solution so that .NET/native interfaces are not chatty

Chatty Interfaces

demo

PDC03

Make the connection

Agenda

- Native and .NET Interop
- Compilation Modes
- Interop Details
- __clrcall Calling Convention
- Mixed DLL Loading
- Delayed CLR Loading
- Application Domains

Managed Calling Convention

- __clrcall keyword
- As a modifier to function
  - Specifies that there is no unmanaged entry point for this function
  - Uses managed vtable fixup on virtual functions
- As a modifier to a function pointer
  - Specifies that the fptr points to a managed function
  - Used to select the managed entry point for an overloaded function
- __declspec(__callingconvention(__clrcall))
- Uses
  - Penalty free virtual calls
  - Specifying /clr:pure on individual types
  - Interop with Pure MSIL externs

__clrcall

demo

PDC03

Make the connection

Agenda

- Native and .NET Interop
- Compilation Modes
- Interop Details
- __clrcall Calling Convention
- Mixed DLL Loading
- Delayed CLR Loading
- Application Domains

Mixed DLL Loading

- **Mixed DLL Loading Problem Solved!**
- Problem
  - .NET code runs under the Loader Lock
- In VS .NET 2003
  - Required /clr DLLs to be linked /NOENTRY
  - Manually initialize DLLs in consumer
- In Whidbey, just rebuild your DLL
- **If you ignored the 2003 warning: Rebuild your DLL!**

Agenda

- Native and .NET Interop
- Compilation Modes
- Interop Details
- __clrcall Calling Convention
- Mixed DLL Loading
- Delayed CLR Loading
- Application Domains

Delayed CLR Loading

- Loading the CLR upfront has downsides
- Mixed Images in VS .NET 2003
  - CLR loads on process startup
- Mixed Images in Whidbey
  - Very small code path in CLR runs on startup
    - MSCOREE.dll
  - Remainder of CLR Load Deferred

Delayed CLR Loading

demo

PDC03

Make the connection

Agenda

- Native and .NET Interop
- Compilation Modes
- Interop Details
- __clrcall Calling Convention
- Mixed DLL Loading
- Delayed CLR Loading
- Application Domains

Application Domains

- Unit of isolation for a managed application
- Multiple ADs can run in a single process
- ADs can be independently stopped
- No direct access of code or data across ADs
- Configuration info is scoped by AD
- Assemblies can be loaded in multiple ADs
- Used in ASP.NET

Native AppDomain APIs

- Use the new CLR Hosting interface
- Interface provides (among other things)
  - Obtain interface for currently hosted CLR
  - Get and set currently running AD
  - Enumerate available ADs
- Choose AD in which a managed call will run
  - ICLRRuntimeHost::ExecuteInDomain
- Native code can obtain an ICLRRuntimeHost for running CLR instance
- Downside – COM is heavyweight for some scenarios

Statics and Globals

- Global variables and static members of native types
- In VS .NET 2003
  - Process globals (RVAs)
  - Shared across AppDomains
  - Violates concept of AppDomain as unit of isolation

Whidbey AppDomain Features

- __declspec(process)
- __declspec(appdomain)
- Default to per AppDomain for /clr and /clr:pure
  - Expected behavior
- Enable per AppDomain statics in CRT
  - .cctor on module
- Native interface for accessing per AppDomain statics
  - Library support

Statics And Globals

demo

PDC03

Make the connection

Summary

- Rewriting existing C++ code to use WinFX is NOT necessary
- C++ CLR and interop features offer best interop option for most users
- Be aware of transition costs when constructing interop solutions
- Mixed DLL Loading Problem Fixed!
- Be mindful of appdomains if necessary

Additional Resources

- Related sessions
  - Targeting the CLR (TLS401) – See Slide Deck
- Send us feedback!
  - srcurrie@microsoft.com
  - vswish@microsoft.com
  - Public Newsgroups
    - microsoft.public.vc.*
    - microsoft.public.dotnet.languages.vc.*
- Other resources
  - http://www.vertigosoftware.com/quake
  - VC++ Hands On Lab Here at the PDC
  - Ask The Experts
  - VC++ Product Booth

Microsoft®

Make the connection

Microsoft Professional Developers Conference 2003

October 26 - 30, 2003, Los Angeles, CA

Microsoft